Winning Space Race with Data Science

Ryan S.
30 December 2023

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

Executive Summary

Summary of methodologies

- Data Collection through API and Web Scraping
- Data Wrangling for outcome variables
- EDA with Visualization techniques
- EDA with SQL
- Use Folium to build an interactive map
- Create a dashboard with Plotly Dash
- Build Models to predict outcomes using classification methods

Results

- Exploratory Data Analysis
- Interactive Analytics (maps based)
- Predictive Analytics (machine learning)

Introduction

Background

Space X advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars due to their ability to reuse the first stage of their rocket. This is a relatively inexpensive cost considering the nearest competition from other providers can cost in excess of $165 million. If we can determine if the first stage will land, we can determine the cost of a launch. This information can provide useful if we wish to determine the price of a launch for an upstart Space Y company. To execute this objective, we employ mining public data and machine learning models to predict whether certain payloads, orbits and location launches can reuse the first stage of the Falcon 9 rocket.

Exploring Solutions

- How multiple variables affect first-stage landing success (orbit, launch site, etc.)
- What is the best way to predict the success of a first-stage landing
- Review and classify successful landings over time

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - *Employ web scraping techniques and data from SpaceX REST API*
- Perform data wrangling
  - *Filter, re-distribute missing values and use one-hot encoding to organize data for eventual analysis and modelling*
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - *How to build, tune, evaluate classification models (Regression, KNN, SVM, DT)*

Data Collection

General Process

- Acquire data using web scraping and SpaceX REST API
- Filter data and handle missing values to prepare it for analysis and modelling
- Use SQL and EDA to explore data
- Apply Folium and Plotly to look at the data – visualize.
- Predict outcomes using classification models with machine learning

Data Collection – SpaceX API

- Request data from SpaceX
- Use .json() to decode and convert with .jsob_normalize
- Create a dictionary and convert to Df
- Replace missing values
- Export to CSV

[Github Link API]

Now let's start requesting rocket launch data from SpaceX API with the following URL:

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
response = requests.get(spacex_url)
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

```
# Use json_normalize meethod to convert the json result into a dataframe
response = requests.get(static_json_url).json()
data = pd.json_normalize(response)
```

Finally lets construct our dataset using the data we have obtained. We we combine the columns into a dictionary.

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

Then, we need to create a Pandas data frame from the dictionary launch_dict.

```
df = pd.DataFrame.from_dict(launch_dict)
```

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

```
# Calculate the mean value of PayloadMass column
mean = data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].fillna(mean)
data_falcon9.isnull().sum()
```

Data Collection - Scraping

- Requested data from Wikipedia
- Create a beautifulSoup object
- Extract column names and collect data
- Create a dictionary
- Create a Df
- Export to a CSV
- [Github Webscrape]()

```
# use requests.get() method with the provided static_url
# assign the response to a object
page = requests.get(static_url)
page.status_code
```

```
200
```

Create a `BeautifulSoup` object from the HTML `response`

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(page.text, 'html.parser')
```

```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

```
column_names = []
temp = soup.find_all('th')
for x in range(len(temp)):
    try:
     name = extract_column_from_header(temp[x])
     if (name is not None and len(name) > 0):
        column_names.append(name)
    except:
     pass
```

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']


launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

```
headings = []
for key,values in dict(launch_dict).items():
    if key not in headings:
        headings.append(key)
    if values is None:
        del launch_dict[key]

def pad_dict_list(dict_list, padel):
    lmax = 0
    for lname in dict_list.keys():
        lmax = max(lmax, len(dict_list[lname]))
    for lname in dict_list.keys():
        ll = len(dict_list[lname])
        if  ll < lmax:
            dict_list[lname] += [padel] * (lmax - ll)
    return dict_list

pad_dict_list(launch_dict,0)

df = pd.DataFrame.from_dict(launch_dict)
df.head()
```

Data Wrangling

- Performed EDA to acquire:
  - Number of launches at sites
  - Number of orbits
  - Occurrence of orbits
  - Mission outcomes per orbit type
- Created landing outcome label
- Delt with null values
- Export results to CSV

[Github - Data Wrangling]

```python
# Apply value_counts() on column LaunchSite
df["LaunchSite"].value_counts()
```

```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

```python
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
Name: Orbit, dtype: int64
```

```python
# landing_outcomes = values on Outcome column
landing_outcomes = df["Outcome"].value_counts()
landing_outcomes
```

```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: Outcome, dtype: int64
```

```python
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for key,value in df["Outcome"].items():
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

```python
df['Class']=landing_class
df[['Class']].head(8)
```

```python
df["Class"].mean()
```

```
0.6666666666666666
```

We can now export it to a CSV for the next section,but to make the answers consistent, in the next lab we will provide data in a pre-selected date range.

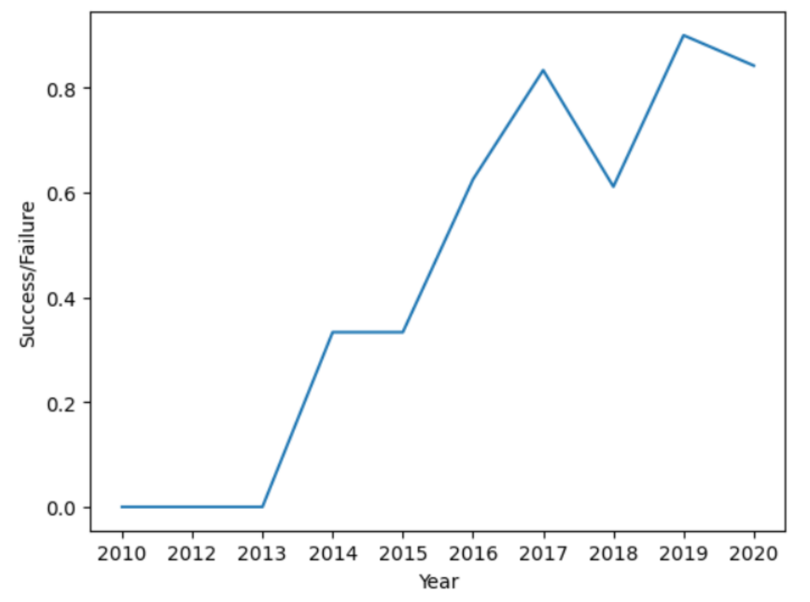
```python
df.to_csv("dataset_part_2.csv", index=False)
```

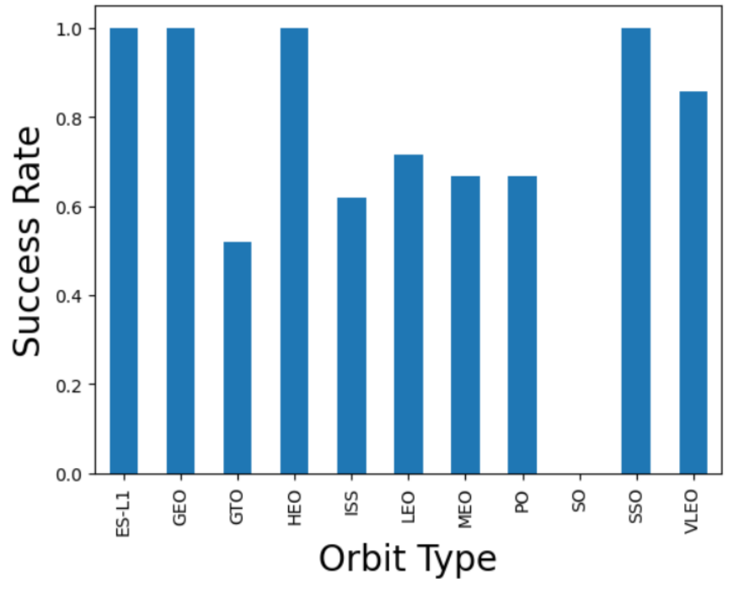
EDA with Data Visualization

Chart Exploration

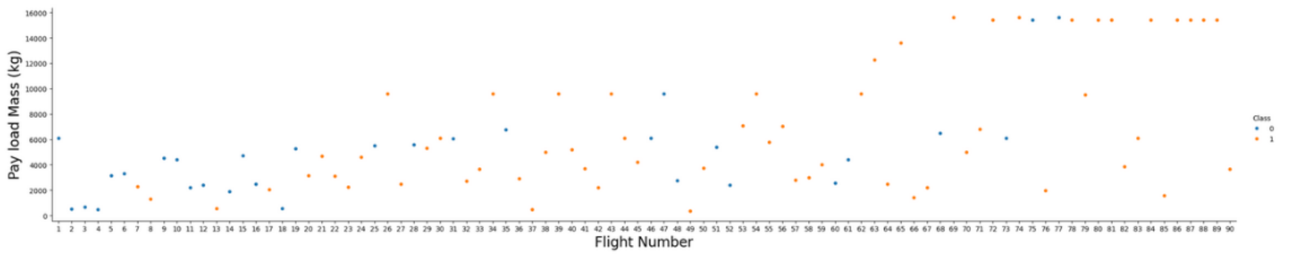
- Flight # v. Payload
- Flight # v. Site
- Payload v. Site
- Payload v. Orbit type

The goal was to view relationships and show comparisons via scatter plots and bar charts






```python
sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```



[Github - EDA with Viz]

EDA with SQL

SQL Queries:

- Display names of unique launch sites in the space mission
- Display 5 records where launch site begins with KSC
- Display total payload mass by boosters launched by NASA
- List the date where successful landing in a drone ship was achieved
- Display average payload mass carried by F9 v1.1
- List the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
- List the records which will display the month names, succesful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017
- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

[Github - EDA with SQL]()

Build an Interactive Map with Folium

Map Objects

- Markers indicating launch sites: Blue (NASA), Red (all other launch sites)
- Markers indicating outcomes: Green (successful), Red (failure)
- Markers were added to help visualize locations and success of launches regarding location and surroundings.
  - The goal was to see which launch sites had the highest success and identify any geographic similarities between sites
- Distances to geographic landmarks (coastline, railway, towns, etc.) were calculated to assess if distance from certain landmarks was a contributing factor to launch site location

Github - Launch sites with Folium

Build a Dashboard with Plotly Dash

We created an interactive dashboard using plotly dash to visualize:

- Pie charts to show total launches by specific launch sites
- Dropdown lists to select single or multiple launch sites
- A slider to allow selection of various payload mass ranges
- Scatter chart visualizing payload mass v. success rate by booster version

These options were utilized so someone could browse search and selection of criteria without having to be specific in their queries.

Github - Dashboard with plotly

Predictive Analysis (Classification)

Summary

- Created a numpy array which them was standardized with StandardScalar to fit and transform the data
- Data was split using train_test_split
- Created a grid search object with a cv equal to 10 (CV=10) to optimize parameters
- GridSearch used on: LR, SVM, decision trees, KNN
- Used accuracy as the metric for modelling, improved the model using feature engineering and algorithm tuning
- ID best model using Jaccard, F1, and accuracy

[Github - Predictive Analysis]()

Create a NumPy array from the column `Class` in `data`, by applying the method `to_numpy()` then assign it to the variable `Y`,make sure the output is a Pandas series (only one bracket df['name of column']).

```
Y = data['Class'].to_numpy()
Y
```

```
array([0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
       1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1], dtype=int64)
```

Standardize the data in `X` then reassign it to the variable `X` using the transform provided below.

```
# students get this
transform = preprocessing.StandardScaler()
X = transform.fit_transform(X)
```

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=2)
```

we can see we only have 18 test samples.

```
Y_test.shape
```

```
(18,)
```

Apply above with GridSearch on regression, SVM, trees, KNN…

Conclude that all methods performed quite closely to each other.

Results

Exploratory data analysis results

Over time, launch success has improved
KSC LC-39A has the best success rate of landing sites
SSO, GEO, HEO, and ES-L1 have 100% success rate

Visual Analytic Results

Launch sites are typically near the equator and near the coast
They are isolated enough that if a catastrophic error were to occur, people and landmarks would not be in danger

Predictive analysis results

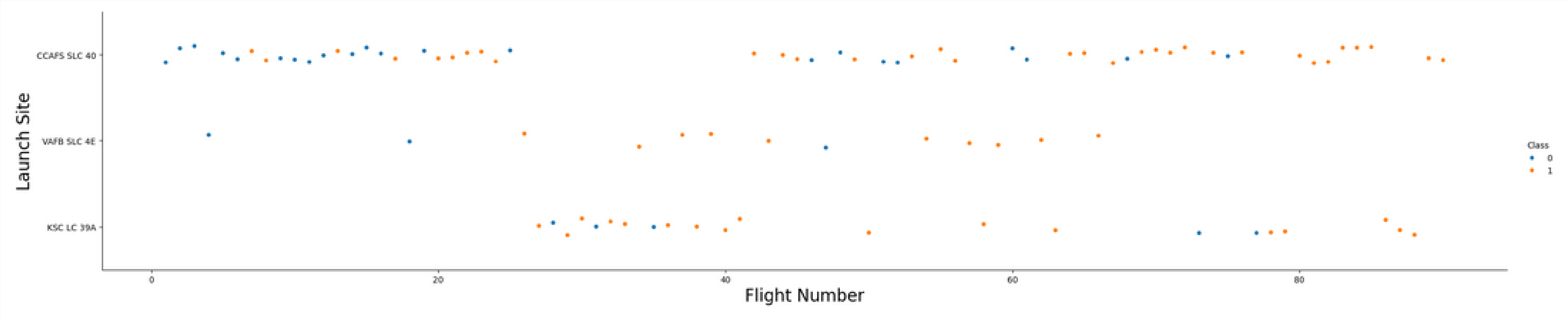
Most models performed very closely, but using a Decision Tree was slightly better as a predictive model with training data, but fit test data slightly worse

Section 2

Insights drawn from EDA

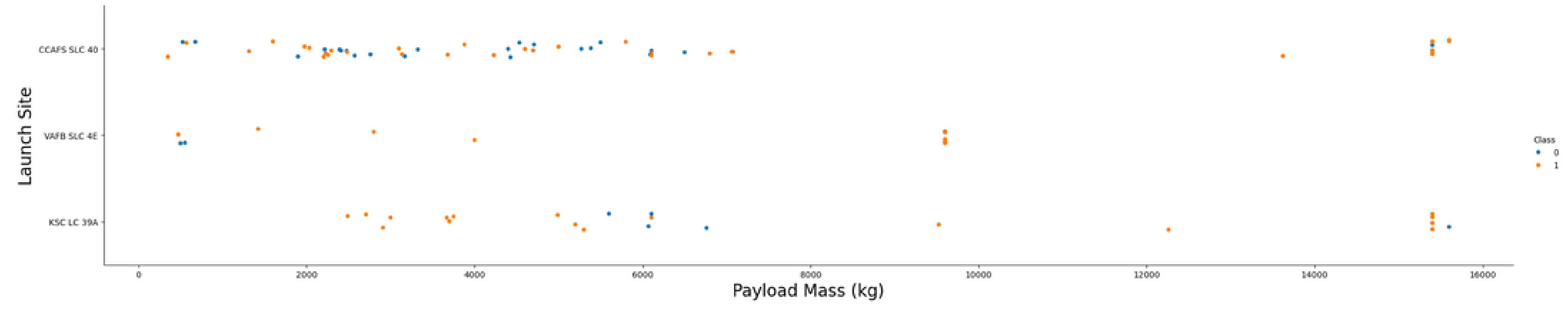
Flight Number vs. Launch Site

- Blue indicates failure, orange is success
- Earlier flights had higher failure rates than later flights
- Most launches occur at CCAFS
- In general, the “newer” the launch the better chance for success across the 3 sites



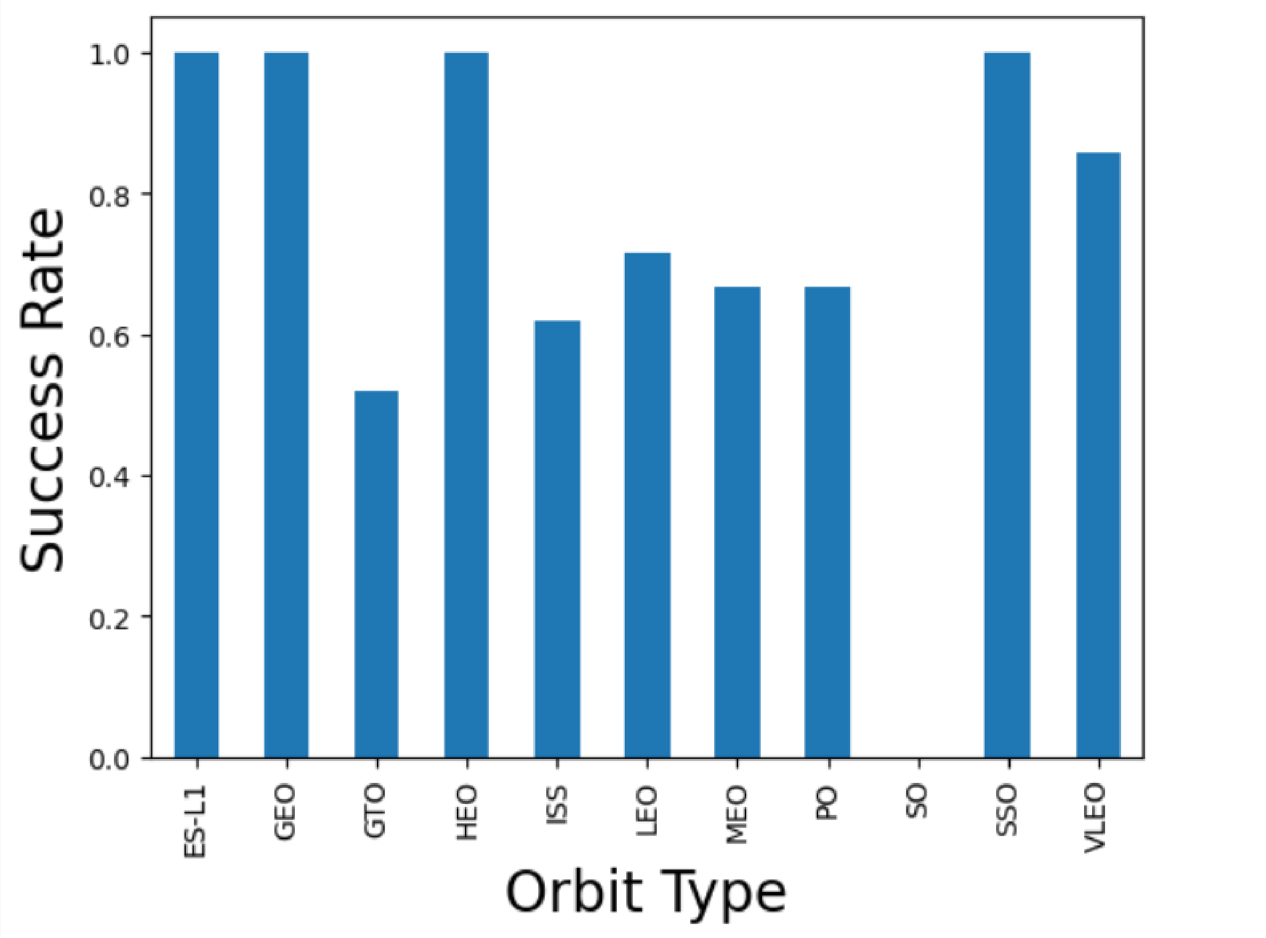
Payload vs. Launch Site

- Blue indicates failure, orange is success
- Higher payload mass leads to greater success
- Most payloads are 7000kg or under



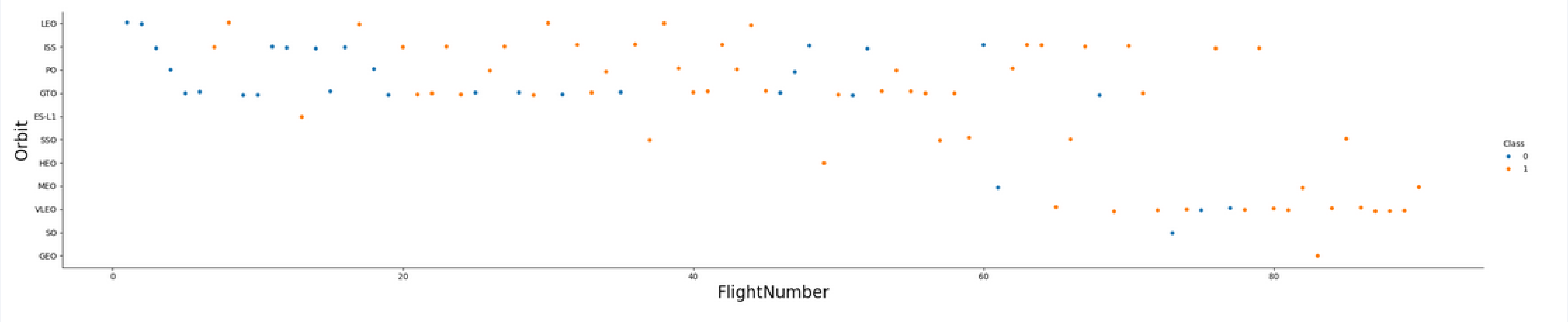
Success Rate vs. Orbit Type

- ES-L1, GEO, HEO, and SSO have 100% success rate
- SO has 0 successful launches
- Most orbits have a middling range of success



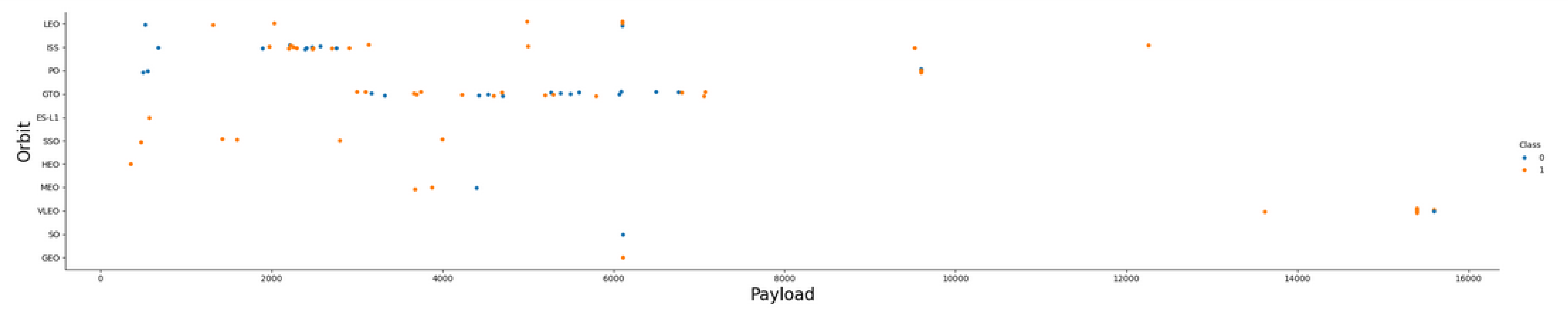
Flight Number vs. Orbit Type

- Blue indicates failure, orange is success
- Later flights had higher opportunity of success
- LEO, ISS, PO, GTO are the oldest orbit usages and taper off as flights become "newer"



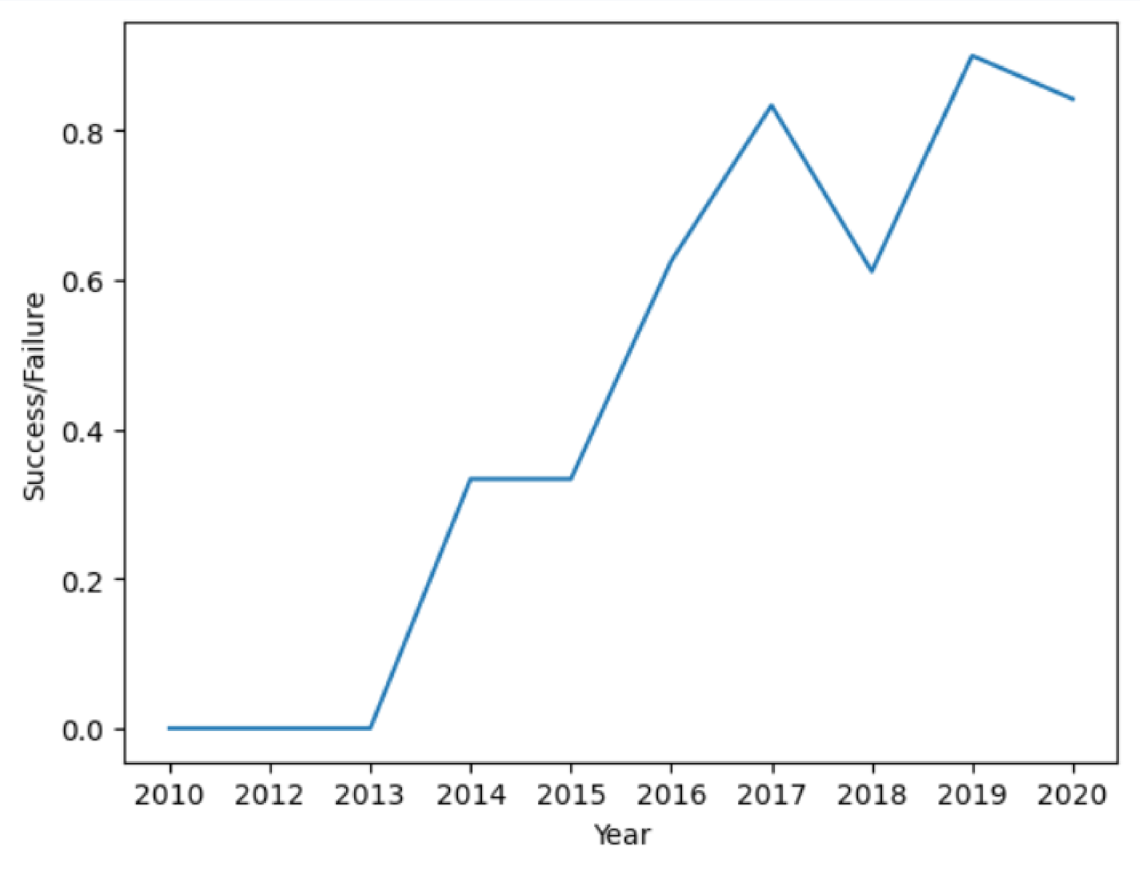
Payload vs. Orbit Type

- Blue indicates failure, orange is success
- GTO is a risky orbit for payloads between 3000-7000kg
- ISS an GTO are the most utilized orbits
- SSO has a 100% success rate for all payload attempts



Launch Success Yearly Trend

Launch success has increased drastically from 2013 to 2020



All Launch Site Names

Query: %sql select distinct(LAUNCH_SITE) from SPACEXTBL

Unique launch sites:

launch_site
CCAFS LC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E

Used “distinct” to acquire the unique launch sites through the query

Launch Site Names Begin with 'KSC'

Limit 5 keeps the query to only amount 5, rather than returning all results

Display 5 records where launch sites begin with the string 'KSC'

```
%sql SELECT LAUNCH_SITE from SPACEXTBL where (LAUNCH_SITE) LIKE 'KSC%' LIMIT 5;
```

```
 * sqlite:///my_data1.db
Done.
```

Launch_Site
KSC LC-39A
KSC LC-39A
KSC LC-39A
KSC LC-39A
KSC LC-39A

Total Payload Mass

Total payload mass for NASA CRS is 45596 based on the below query

Display the total payload mass carried by boosters launched by NASA (CRS)

```
%sql select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where CUSTOMER = "NASA (CRS)"
```

```
 * sqlite:///my_data1.db
Done.
```

sum(PAYLOAD_MASS_KG_)
45596

Average Payload Mass by F9 v1.1

The average payload mass carried by the F9 v1.1 was ~2,928kg

Display average payload mass carried by booster version F9 v1.1

```
%sql select avg(PAYLOAD_MASS__KG_) as payloadmass from SPACEXDATASET;
```

First Successful Ground Landing Date

The first successful landing on a drone ship was April 8th 2016

List the date where the succesful landing outcome in drone ship was acheived.

Hint:Use min function

```
%sql select min(DATE) from SPACEXTBL where Landing_Outcome = "Success (drone ship)";
```

```
 * sqlite:///my_data1.db
Done.
```

min(DATE)
2016-04-08

Successful Drone Ship Landing with Payload between 4000 and 6000

Boosters successfully landing on drone ship: B1022, B1206, B1021.2, B1031.2

List the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000

```
%sql select BOOSTER_VERSION from SPACEXTBL where LANDING_OUTCOME='Success (drone ship)' and PAYLOAD_MASS__KG_ BETWEEN 4000 and 6000;
```

```
 * sqlite:///my_data1.db
Done.
```

Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

Total number of outcomes is 99

List the total number of successful and failure mission outcomes

```
%sql select count(MISSION_OUTCOME) as MissionOutcomes from SPACEXTBL where MISSION_OUTCOME='Success' or MISSION_OUTCOME='Failure (in flight)';
```

```
 * sqlite:///my_data1.db
Done.
```

MissionOutcomes
99

Boosters Carried Maximum Payload

We need to search category booster version, but only find MAX values for payload masses in that category

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
%sql select BOOSTER_VERSION from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTBL);
```

```
 * sqlite:///my_data1.db
Done.
```

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

2015 Launch Records

Most results of query are from launch site KSC LC-39A. We needed to use “as” and “where” to re-name fields and indicate what outcomes we were searching for.

```
%sql SELECT substr(Date,6,2) as month,substr(Date,9,2) as DATE,MISSION_OUTCOME,BOOSTER_VERSION,LAUNCH_SITE FROM SPACEXTBL where Landing_Outcome='Success (ground pad)' and substr(Date,0,5)='2017'
```

```
 * sqlite:///my_data1.db
Done.
```

month	DATE	Mission_Outcome	Booster_Version	Launch_Site
02	19	Success	F9 FT B1031.1	KSC LC-39A
05	01	Success	F9 FT B1032.1	KSC LC-39A
06	03	Success	F9 FT B1035.1	KSC LC-39A
08	14	Success	F9 B4 B1039.1	KSC LC-39A
09	07	Success	F9 B4 B1040.1	KSC LC-39A
12	15	Success	F9 FT B1035.2	CCAFS SLC-40

Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

- Used “where” and “between” to identify date range
- Used “group by” and “order by” “desc” (descending) to sort the data

```
%sql SELECT [Landing_Outcome], count(*) as count_outcomes from SPACEXTBL WHERE DATE between '2010-06-04' and '2017-03-20' group by [Landing_Outcome] order by count_outcomes DESC;
```

```
 * sqlite:///my_data1.db
Done.
```

Landing_Outcome	count_outcomes
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

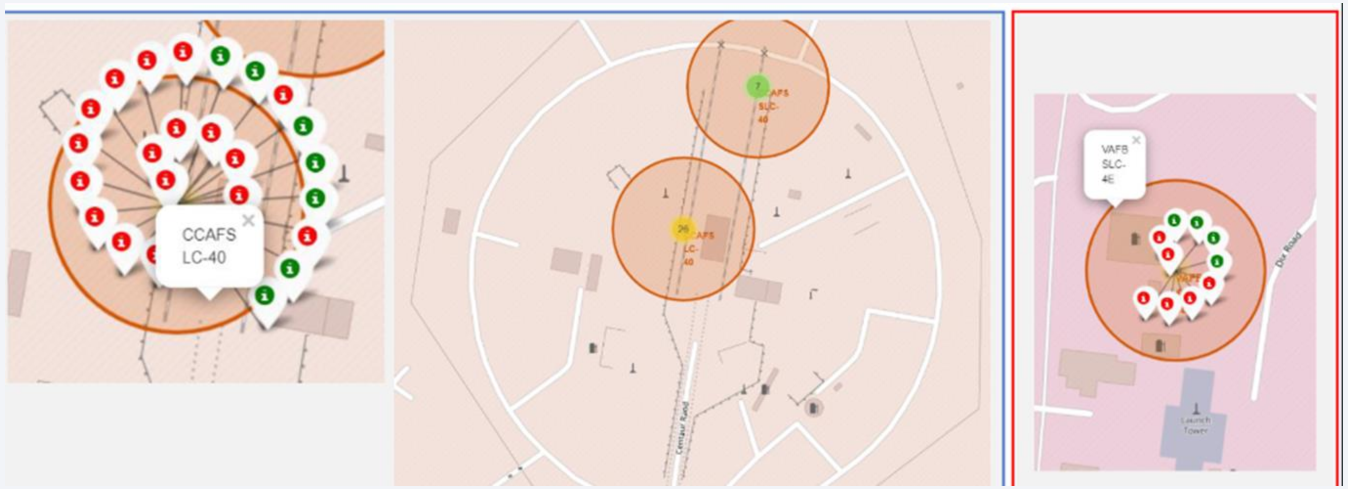
Global View of Launch Locations

Launch sites share similar characteristics:

- Located along the coast
- Located near the equator to make orbital trajectories easier to attain

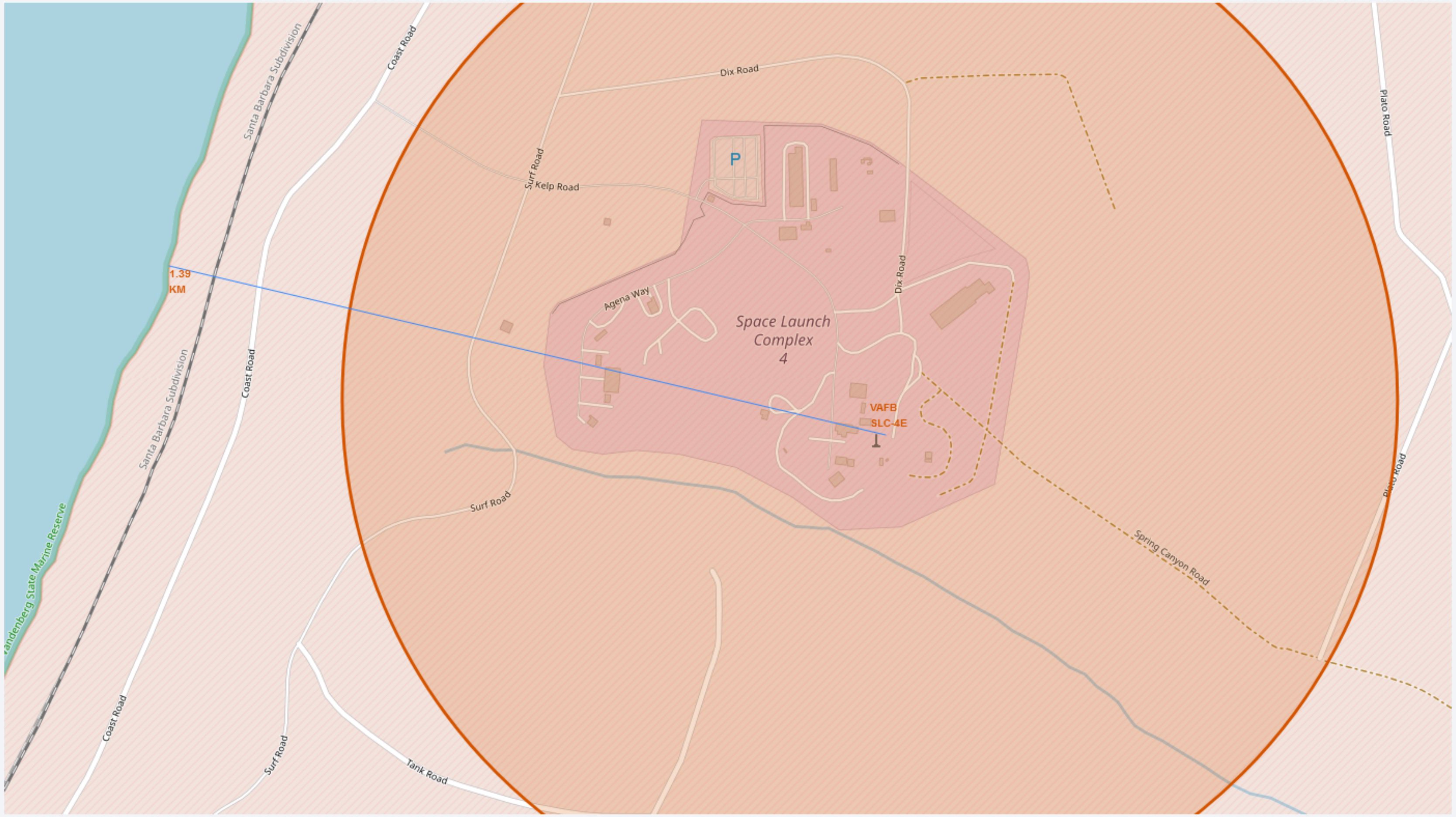
Successful/Unsuccessful Launch by Location

- Red markers are failures, green markers are success
- There have been many more launches from Florida (left) than California (right)



Launch Complex Proximity to Landmarks

- VAFB SLC-4E site is 1.39km from the coast
- It has multiple road access points to ease equipment shipping and transfer
- It is not located near towns, major highways, or railways

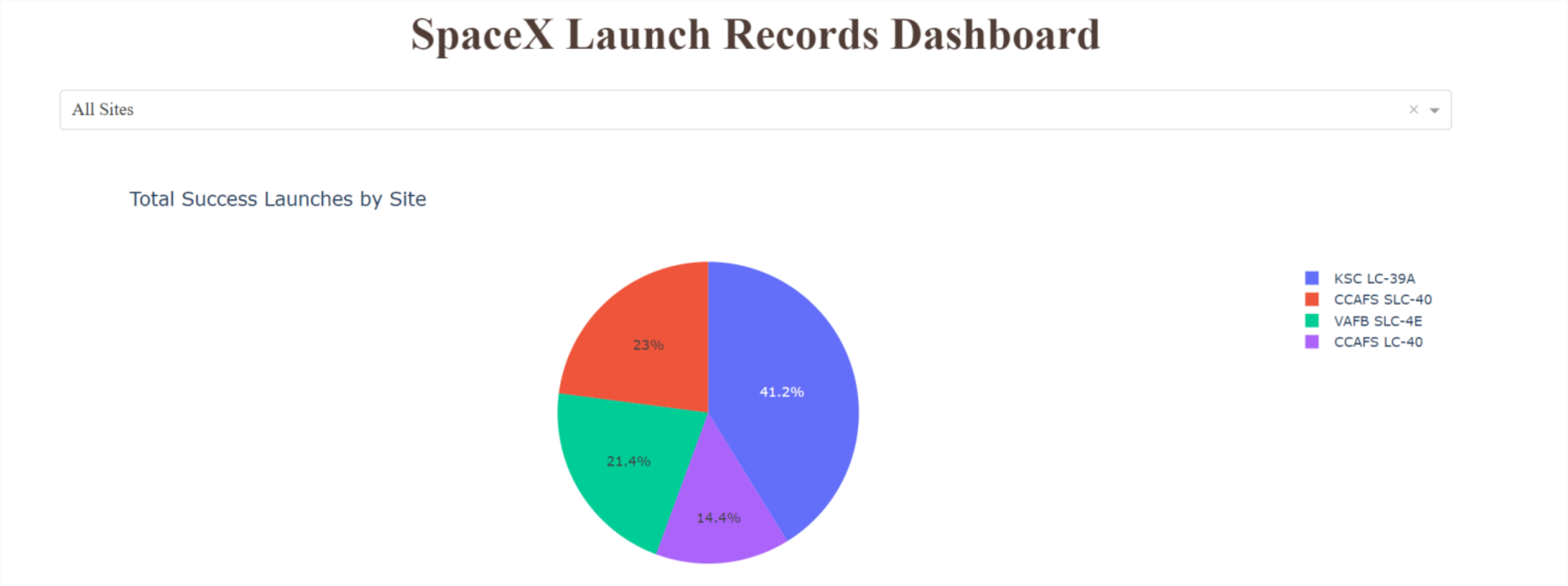


Section 4

Build a Dashboard with Plotly Dash

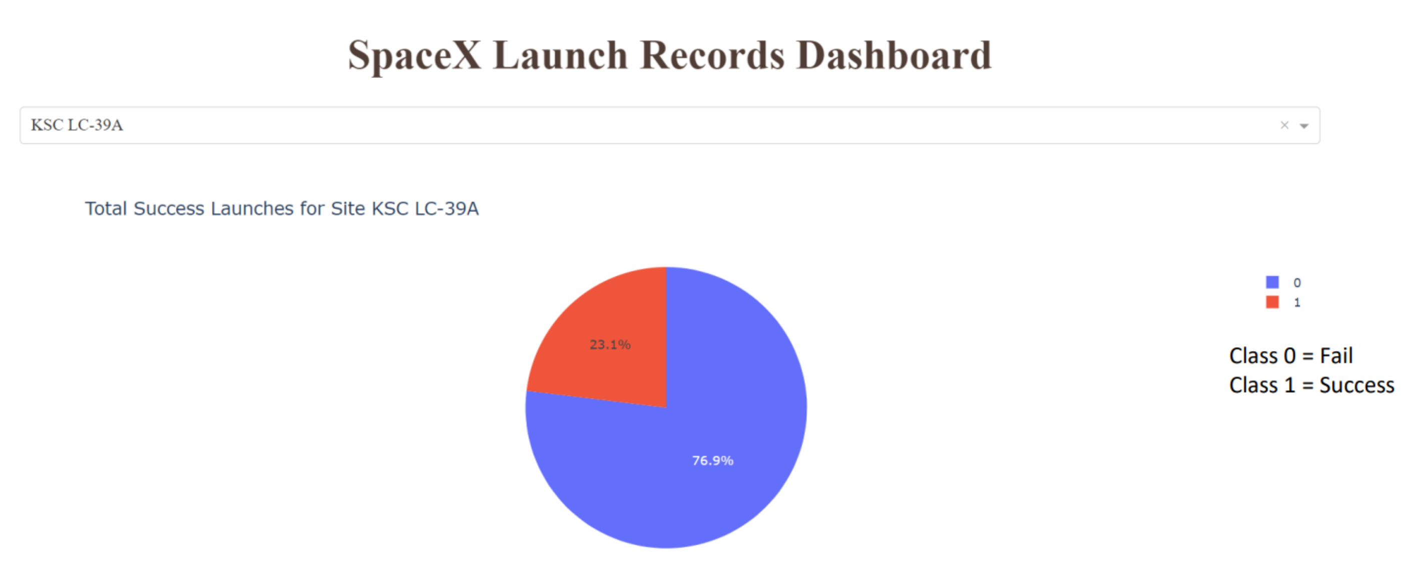
Success count for all Sites

KSC has the most total successful launches



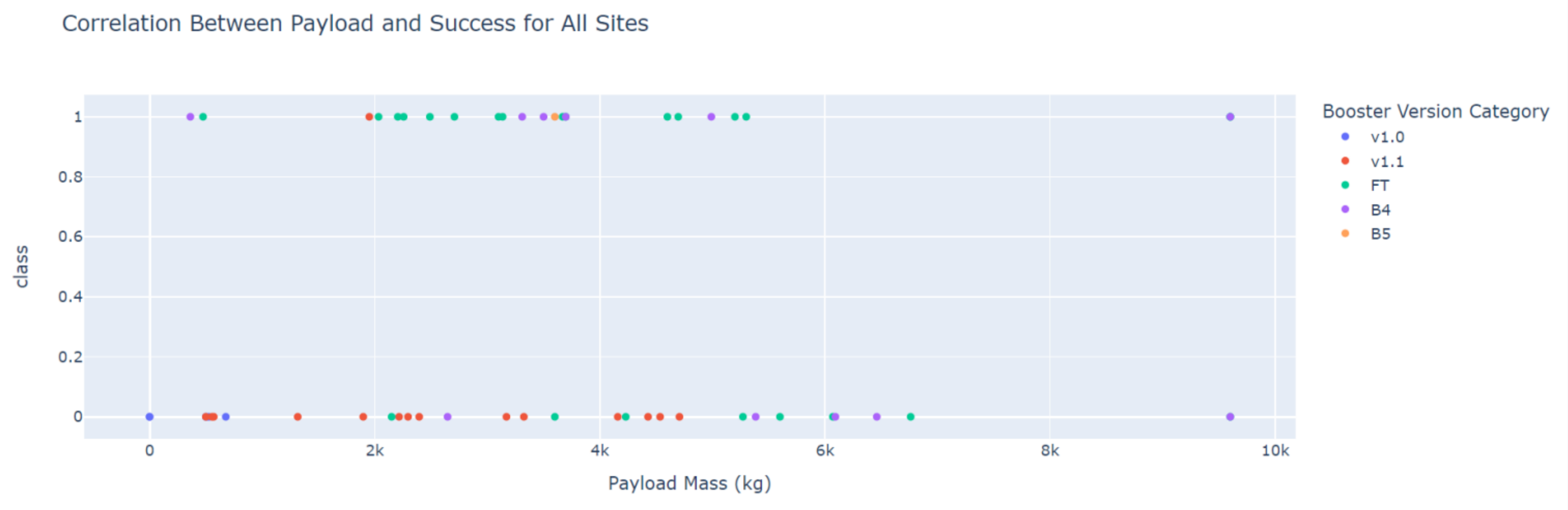
Launch site: Highest success ratio

KSC had a nearly 77% launch success rate, the highest among launch sites



Payload v. Launch outcome: All site comparison

Lower payloads typically had a higher chance of success across all sites



Section 5

Predictive Analysis (Classification)

Classification Accuracy

Decision tree model has the best accuracy with a score of 0.873

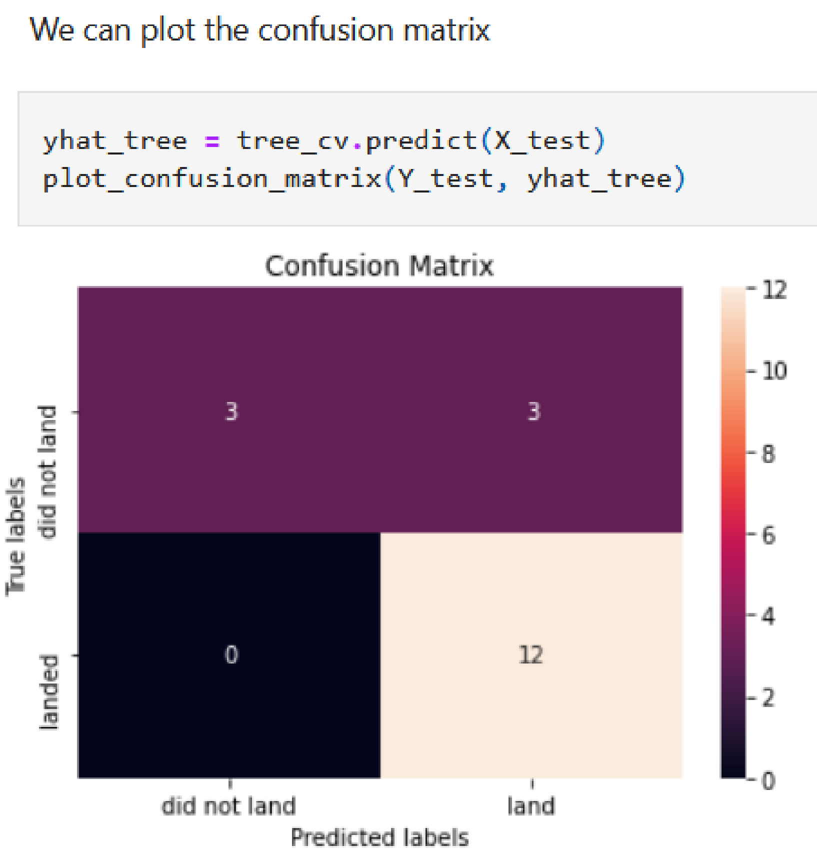
Find the method performs best:

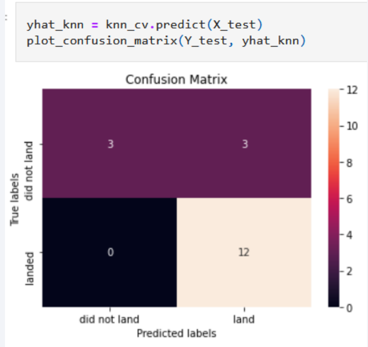
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

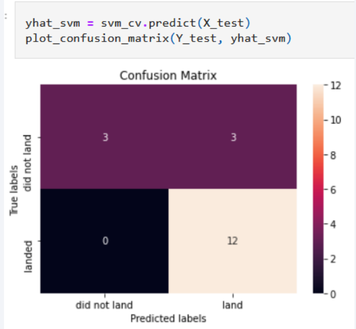
```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split
': 5, 'splitter': 'random'}
```

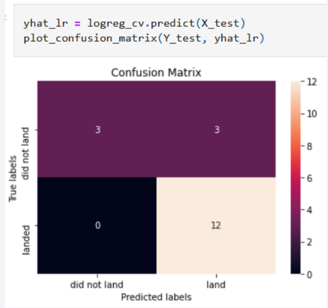
Confusion Matrix

- This matrix summarizes the performance of a classification algorithm
- False positives do occur
- All confusion matrix results for these comparisons were the same across machine learning methods









Conclusions

General takeaways from data analysis:

- Location: launch sites are near the coast at the equator and avoid major highways, cities, and railways
- Experience: as SpaceX experience grows, launch success has seen a consistent increase over time
- Success: KSC is the best launch site, and excels at launches that have a weight that is under 6000kg
- Orbital Options: HEO, SSO, GEO, and ES-L1 have a 100% success record. All other orbital trajectories perform significantly worse
- Payload Mass: the higher the mass, the better chance for success.

Thank you!